Claims

1. A methodology for designing a software system independent of a target hardware implementation, the methodology comprising designing the software system as comprising

a first component for realizing a predetermined functionality,

a first coordinator for managing interactions between the first component and a second component, and

a first pair of coordination interfaces comprising a first and a second coordination interface for implementing a connection between the first component and the first coordinator so as to preserve component modularity while exposing only the parts of the component that participate in coordination.

2. A method according to claim 1 wherein the first coordinator implements a predetermined coordination protocol.

3. A method according to claim 2 wherein the first pair of coordination interfaces includes a pair of complimentary ports to transfer information between the coordination interfaces.

4. A method according to claim 3 wherein a first port of the pair of complimentary ports has a combination of attributes realizing an output message port and the other port of the pair of complimentary ports has a combination of attributes realizing an input message port.

5. A method according to claim 3 wherein the a first port of the pair of complimentary ports has a combination of attributes realizing an exported state port and the other port of the pair of complimentary ports has a combination of attributes realizing an imported state port.

6. A method according to claim 3 wherein a first port of the pair of complimentary ports has a combination of attributes realizing a first control port, and the other port of the pair of complimentary ports has a combination of attributes realizing a second control port that is complimentary to the first control port.

7. A method according to claim 3 wherein a first port of the pair of complimentary ports has a combination of attributes realizing a first arbitrated state port, and the other port of the pair of complimentary ports has a combination of

attributes realizing a second arbitrated state port that is complimentary to the first arbitrated state port.

8. A method according to claim 3 wherein each port of the pair of complimentary ports is defined by a five-tuple (T, A, Q, D, R) where:

T represents a datatype of the port;

A is a Boolean value that is true if and only if the port is arbitrated;

Q is a predetermined integer greater than zero that represents logical queue depth of the port;

D represents a directionality of data flows with respect to the port; and

R represents a policy for data removal on the port.

9. A method according to claim 8 wherein D is one of { in, out, inout } representing data flow into the port, out of the port, or bi-directional, respectively.

10. A method according to claim 9 wherein D is one of { in, out, inout or custom}, where custom directionality permits restricting the port to accept or to generate only certain specific predetermined values.

11. A method according to claim 2 wherein the first coordination interface implements a predetermined guarantee of a selected invariant interface property of the first component.

12. A method according to claim 11 wherein the guarantee specifies a predetermined event ordering.

13. A method according to claim 11 wherein the guarantee specifies an acceptable relative behavior between the first and second component.

14. A method according to claim 2 wherein the first coordination interface includes a specified requirement and the second coordination interface includes a specified guarantee that satisfies the specified requirement of the first coordination interface.

15. A computer software design methodology comprising a coordination interface for software packaging, wherein the coordination interface comprises at least one named port.

16. A method according to claim 15 wherein the coordination interface further comprises:

a set of at least one named guarantee provided by the interface;

a set of at least one named requirement that must be matched by a guarantee of a connected interface

17. A method according to claim 16 wherein the coordination interface further comprises a set of coordination interfaces including at least a second coordination interface, thereby making coordination interfaces hierarchical.

18. A method according to claim 15 wherein the named port is defined by a five-tuple (T, A, Q, D, R) where:

T represents a datatype of the port;

A is a Boolean value that is true if and only if the port is arbitrated;

Q is a predetermined integer greater than zero that represents logical queue depth of the port;

D represents a directionality of data flows with respect to the port; and

R represents a policy for data removal on the port.

19. A software system for execution on a hardware platform, the software system comprising:

a first component having a first coordination interface;

a second component having a second coordination interface; and

a coordinator for coordinating control and data flow between the first and second components and having a third coordination interface that is complimentary to the first coordination interface and a fourth coordination interface that is complimentary to the second coordination interface.

20. A software system according to claim 19 wherein the first coordination interface includes a first port as a connection point to a second port included on the third coordination interface.

21. A method according to claim 20 wherein the first port has a combination of attributes realizing a first message port.

22. A method according to claim 21 wherein the second port has a combination of attributes realizing a second message port that is complimentary to the first message port.

23. A method according to claim 20 wherein the first port has a combination of attributes realizing a first state port.

24. A method according to claim 22 wherein the second port has a combination of attributes realizing a second state port that is complimentary to the first message port.

25. A method according to claim 20 wherein the first port has a combination of attributes realizing a first control port.

26. A method according to claim 25 wherein the first port has a combination of attributes realizing a second control port that is complimentary to the first control port.

27. A method for designing software systems comprising:

designing a first software component for performing a first predetermined functionality that when activated produces a defined result;

designing a first coordination interface for logically connecting the first component to the first coordination interface in order to export the result produced by the first component;

designing a second component for performing a second predetermined functionality that can respond to the defined result produced by the first component;

designing a second coordination interface for logically connecting the second component to the second coordination interface in order to importing the result produced by the first software component;

designing a coordinator with a third coordination interface that is complimentary to the first coordination interface and a fourth coordination interface that is complimentary to the second coordination interface, for transferring the result exported by the first coordination interface from the first coordination interface to the second coordination interface.

28. A method according to claim 27 in which the first coordination interface has a first port for exporting the result produced by the first component.

29. A method according to claim 28 in which the second coordination interface has a second port for importing the result produced by first component.

30. A method according to claim 29 in which the third coordination interface has a third port that is complimentary to the first port for importing the result exported by the first port, and in which the fourth coordination interface has a fourth port that is complimentary to the second port for exporting the result imported by the third port.

31. A method according to claim 30 in which the third coordination interface is bound to the fourth coordination interface in a manner that implements a predetermined coordination protocol.

32. A method according to claim 27 in which the first component comprises a first action, for implementing the first predetermined functionality, a first mode for implementing a boolean guard on the first action, and a first event to serve as a trigger for initiating the first action.

33. A method according to claim 32 in which the second component comprises a second action, for implementing the second predetermined functionality, a second mode for implementing a boolean guard on the second action, and a second event to serve as a trigger for initiating the second action.

34. A method according to claim 33 in which the coordinator comprises a binding between the third and fourth coordination interfaces for transferring the result from the third coordination interface to the fourth coordination interface.

35. A method according to claim 33 in which the coordinator comprises a coordinator action which performs a predetermined coordinator function, and a mode which serves as a boolean guard on the coordinator action.

36. A method according to claim 35 in which the coordinator further comprises a constraint which serves to enforce a predetermined relationship between a pair of control ports.

37. A method for designing a software program without reference to a target system architecture comprising:

creating a first component comprising software code for performing a first function and a first coordination interface for sending and receiving separate control and dataflow information;

creating a second component comprising software code for performing a second function and a second coordination interface for sending and receiving separate control and dataflow information;

creating a coordinator to manage control interactions and dataflow interactions between the first component and the second component comprising a third coordination interface for logically connecting to the first coordination interface, and a fourth coordination interface for logically connecting to the second coordination interface.

38. The method of claim 37 in which the coordination interfaces each comprise a port for transferring information.

39. The method of claim 38 in which the first coordination interface and the third coordination interface have corresponding ports and complimentary requirements and guarantees, and the second coordination interface and the fourth coordination interface have corresponding ports and complimentary requirements and guarantees.

40. The method of claim 39 in which the first component further comprises a first mode and a first action.

41. The method of claim 40 in which the second component further comprises a second mode and a second action.

42. The method of claim 41 in which the coordinator further comprises a third action and a third mode.

43. The method of claim 42 in which the coordinator further comprises a first binding which connects a first port on the third coordination interface to a second port on the fourth coordination interface

44. The method of claim 43 in which the first port is an input data port and the second port is an output data port.

45. The method of claim 42 in which the coordinator further comprises a second binding which connects a third port on the third coordination interface to the third mode.

46. The method of claim 42 in which the coordinator further comprises a third binding which connects a fourth port on the fourth coordination interface to a first variable.

47. The method of claim 42 in which the coordinator further comprises a fourth binding which connects a fifth port on the third coordination interface to a first event.

48. A software system comprising:

a first software component comprising a first action;

a second software component comprising a second action ; and

a coordinator for implementing a communication protocol between the first software component and the second software component and connected to the first software component and the second software component.

49. The software system of claim 48 in which the first software component further comprises:

a first mode connected to the first action;

a first trigger connected to the first action; and

a first coordination interface comprising a first port.

50. The software system of claim 49 in which the second software component further comprises;

a second mode connected to the second action;

a first trigger connected to the first action; and

a second coordination interface comprising a second port.

51. The software system of claim 50 in which the coordinator comprises:

a third action for implementing the communication protocol;

a third mode connected to the third action;

a third trigger connected to the third action;

a third coordination interface comprising a third port and connected to the first coordination interface; and

a fourth coordination interface comprising a fourth port and connected to the second coordination interface.

52. The software system of claim 51 in which the first software component further comprises a first binding.

53. The software system of claim 52 in which the second software component further comprises a second binding.

54. The software system of claim 53 in which the coordinator further comprises a third binding.

55. The software system of claim 54 in which the first coordination interface further comprises a fourth port and a fourth binding.

56. The software system of claim 55 in which the first binding connects the first port to the first mode and the fourth binding connects the fourth port with the first trigger.

57. A method for designing a software system comprising:

creating a first software component comprising a first action, a first mode and a first coordination interface;

creating a second software component comprising a second action, a second mode and a second coordination interface; and

creating a coordinator, to implement a predetermined communication protocol between the first and second component by coordinating control and dataflow interactions between the first component and the second component, comprising:

a third action;

a third mode;

a third coordination interface connected to the first coordination interface; and

a fourth coordination interface connected to the second coordination interface.

58. A software system comprising;

n components, where n is an integer greater than zero, each component designed to perform a predetermined functionality;

m coordinators, where m is an integer greater than zero, each coordinator designed to implement a predetermined coordination protocol between a set of the n components;

n coordination interface pairs, each coordination interface pair designed to logically connect a component to a coordinator for transferring information between the component and the coordinator.

59. The software system of claim 58 wherein each component of the n components comprises:

an action to perform a predetermined function;

a mode to act as a boolean guard on the action;

a trigger which when activated causes the action to perform the predetermined function when the mode has a value of true.

60. The software system of claim 59 wherein a component of the n components further comprises:

a set of sub-components;

a set of internal coordinators, each internal coordinator designed to implement a predetermined coordination protocol between a subset of the set of sub-components;

a set internal coordination interface pairs for logically connecting the sub-components to the internal coordinators.

61. The software system of claim 59 wherein each coordinator of the m coordinators comprises:

an action an action to perform a predetermined function;

a mode to act as a boolean guard on the action;

a trigger which when activated causes the action to perform the predetermined function when the mode has a value of true.

62. The software system of claim 61 wherein a coordinator of the m coordinators further comprises:

a set of sub-coordinators, each sub-coordinator designed to implement a predetermined coordination protocol between a subset of any of the n coordination interface pairs that are logically connected to the coordinator;

a set of internal coordination interface pairs, each internal coordination interface pair designed to logically connect the sub-coordinators to one coordination interface of the coordination interface pairs that are logically connected to the coordinator.